Remarks

Amendments Offered

Amendments are hereby offered to

a) amend claim 1 to incorporate the limitations previously presented in claim 3, and correct a previously undetected informality;

b) cancel claim 3;

c) amend claims 4, 7, 12, 21 and 23 to correct a number of previously undetected informalities;

d) amend claim 19 to incorporate the limitations previously presented in claim 20;

e) amend claim 20 to incorporate the limitations previously presented in claim 10.

Since the amendments merely address informality issues, or have been previously presented, they do no introduce new matters, nor raise new issues. Entry of the amendments is respectfully requested to place the claims in condition for allowance or appeal.

Section 112 first paragraph rejections

The Examiner asserted that by way of the amendments entered in the last response, Applicant introduced new matters. Specifically, the amended language of "second obscuring instructions" of claims 1, 10 and 19, and the "runtime manager" of claims 6, 20 and 27 constitute new matters. Applicant respectfully disagrees.

With respect to language of "second obscuring instructions", Applicant directs the Examiner's attention to the specification,

a) page 7, lines 27-29, where it is disclosed "Obscuring code bank 204 is a database that contains program instructions ("first obscuring instructions") previously created through a manual, automated, or in combination of manual and automated process" (annotation added);

b) page 7, lines 20-22, where it is disclosed "obscuring code generator 203 generates obscuring code blocks 206 ("second obscuring instructions") that are used to protect the critical function source code 101" (annotation added);

c) page 8, lines 2-4, where it is disclosed "each code block 206 ("second obscuring instructions") is a subset of the obscuring instructions available in obscuring code bank 204 ("first obscuring instructions")" (annotation added).

Applicant also invites the Examiner to view Figure 2.

Accordingly, the language of "first and second obscuring instructions" is clearly supported by the specification.

With respect to the language of "runtime manager", Applicant directs the Examiner's attention to the specification,

a) page 12, lines 24-26, where it is disclosed "obscuring code injector 301 combines the serialized code blocks 104 and obscuring code blocks 206 with runtime apparatus 302 to create a pre-compilation obscured image 307" (underline added);

b) page 12, lines 28-30, where it is disclosed "Runtime apparatus 302 comprises the necessary programming instructions to load blocks of machine level code into computer's memory for execution and to transfer execution control from one code block to another" (underline added).

The functions of "instruction loading", "execution control transfer", and other functions of the like are commonly referred to as "runtime or execution time management functions" by those skilled in the computer art.

Accordingly, the language of "runtime manager" is clearly supported by the specification.

Therefore, it is clear that no new matters have been introduced. Thus, Applicant respectfully requests the rejections be withdrawn.

Section 102(b) rejections

The Examiner maintained that the 102(b) rejections against amended claims 1-5, 19 and 21, asserting that all limitations are anticipated by Aucsmith. Applicant respectfully disagrees.

Aucsmith is directed towards tamper resistant methods for protecting a security sensitive program that operates with a secret. Aucsmith teaches having the secret be distributed in time and space (see e.g. Abstract). Specifically, the distribution may be accomplished by partitioning the security sensitive program into subprograms, and the secret into portions, and distributing the different secret portions in the different subprograms (see e.g. Fig. 1). The subprograms may be further interleaved with unrelated tasks (col. 4, lines 16-20).

Aucsmith further teaches protecting the partitioned arrangement (with or without interleaving unrelated tasks) with obfuscation (see e.g. Abstract). Specifically, the subprograms (except the initial subprogram where execution starts) may be encrypted (using various keys), and associated with each other in accordance with a selected pattern of mutation (see e.g. Abstract), where the pattern of mutation, at each mutation, recovers the next subprogram to be executed into its plain executable state (from the encrypted state) (see e.g. Fig. 14). An

example of mutation is the performance of the XOR operation with a designated partner subprogram (see e.g. Fig. 14). Accordingly, for the example embodiment, each subprogram, in addition to the original subprogram code and a portion of the secret, is provided with the mutation partner identification function, the mutation function, the partner key and the jump instruction with the starting address of the next subprogram (see Fig 5).

Turning now to claim 1, which on entry of the offered amendments, will include the limitations previously presented in claim 3, including in particular, the limitations of

> transforming a first set of obscuring instruction identification codes associated with some or all of the first obscuring instructions to generate a second set of obscuring instruction identification codes;
> generating second obscuring instructions using the second set of obscuring instruction identification codes;

Note that claim 1 includes both the term "instruction" and the term "code", in particular the usage of the term "code" is qualified as "obscuring instruction identification code". Thus any one of ordinary skill in the art would clearly understand the phrase "obscuring instruction identification code" to mean "an identifier of the obscuring instruction", and not read "code" as "instruction" (ignoring the descriptive words "obscuring instruction identification" before it). Further, hereinafter, for ease of discussion, these limitations will be referred to as the Transformation and Generating limitations.

As described earlier, when it comes to "obscuring instructions", Aucsmith merely teaches "interleaving the subprograms with unrelated tasks (obscuring instructions)" (col. 4, lines 16-20, underline and annotations added). Aucsmith does not teach or suggest the unrelated tasks (obscuring instructions) as having ***associated identification codes***. Therefore, it follows that Aucsmith does not teach or suggest the required Transformation and Generating limitations.

In paragraph 9 of the office action, the Examiner appears to be making the argument that all "added" instructions in Aucsmith are considered "obscuring instructions", not just the "purposeless" instructions. Applicant submits this is contrary to the plain meaning of the term "obscuring instructions" as understood by those of ordinary skill in the art, as well as how the term is used in the specification.

Nonetheless, even if we adopt the Examiner's interpretation (which Applicant disagree), by virtue of the fact that Aucsmith did not teach the mutation function et al instructions as having "identification codes", Aucsmith still does not teach or suggest the Transformation and Generation limitations.

In rejecting claim 3, the Examiner cites Ausmith's teaching of the employment of a mutation cycle as having anticipated the required Transformation and Generation limitations. As described earlier, the mutation process is employed to successively recover each next subprogram to be executed, in the plain executable state, from its encrypted state. The mutation process has nothing to do with generating additional "obscuring instructions" from an initial pool of "obscuring instructions", where the generated "obscuring instructions" are injected into the instructions to be protected to obscure the instructions to be protected. In particular, the mutation process taught involves performing XOR operations on partner subprograms, and has nothing to do with accomplishing such generation by transforming the ***identification codes*** associated with the initial pool of "obscuring instructions". [Note again, the limitation recites "*transformation of the obscuring instruction identification codes*", not "*transformation of code*" nor "*transformation of instructions*".]

Accordingly, amended claim 1 (which is previously presented claim 3) is clearly patentable over Aucsmith under sec 102(b).

Claims 2 and 4-5 depend on claim 1, incorporating its limitations. Therefore, for at least the same reasons, claims 2 and 4-5 are patentable over Aucsmith under sec 102(b).

Claim 19 has been amended to incorporate the limitations previously presented in claim 20, which includes the limitations of ***injecting a plurality of copies of a runtime manager*** into the instructions to be protected. Note that the language calls for *multiple copies of a runtime manager.* Therefore, these copies are *identical copies of the same runtime manager.*

Claim 19 (previously presented as claim 20) was rejected in view of Aucsmith and Pendakur combined. Accordingly, the Examiner has already conceded that Aucsmith does not teach the injection of a runtime manager. Therefore, claim 19 (previously presented as claim 20) is patentable over Aucsmith under sec 102(b).

The Examiner argued that Pendakur teaches a runtime manager. Therefore, one of ordinary skilled in the art, "motivated by making the runtime manager secure and tamper resistant would include the runtime manager". However, if a runtime manager is the object of the protection, it is already accounted for by the recitation of the instructions receiving the protective "obscuring instructions". If so, what is the motivation of including it again? Further, what is the motivation for including ***multiple copies of the same runtime manager***?

Accordingly, Applicant submits claim 19 (previously presented as claim 20) is patentable over Aucsmith even when combined with Pendakur.

Claim 21 includes in substance the same Transformation and Generating limitations earlier discussed for amended claim 1. Therefore, for at least the same reasons, claim 21 is patentable over Aucsmith under sec 102(b).

103 Rejections

Claims 6-7, 20, 24 and 27 were rejected in view of Aucsmith combined with Pendakur.

Claims 6-7, 20 and 24 depend on either claim 1 or 19, incorporating its limitations. Therefore, for at least the same reasons, claims 6-7, 20 and 24 are patentable over Aucsmith.

Pendakur does not cure the earlier discussed deficiencies of Aucsmith. Therefore, claims 6-7, 20 and 24 are patentable over Aucsmith even when combined with Pendakur.

Claim 27 depends on claim 21 incorporating its limitations. Therefore, claim 27 is patentable over Aucsmith. Pendakur does not cure the earlier discussed deficiencies of Aucsmith. Therefore, claim 27 is patentable over Aucsmith even when combined with Pendakur.

Claims 10-13, 17 and 22-23 were rejected in view of Aucsmith combined with Bellare (US5892899). In rejecting claim 10, the Examiner asserted that Bellare remedied Aucsmith's failure to teach "nested encrypting". Specifically, the Examiner relied on the disclosure in col. 1, lines 48-60, col. 2, lines 42-45 and claim 3 of Bellare.

Col.1, lines 48-60 merely teaches the first cipher block as having the first plain text, the second cipher block as having the first and second plain text, the third cipher block as having the first, second and third plain text, and so forth. Thus, the passage merely teaches the ciphering blocks as having overlapping contents, and they are not nested, as the term is understood by those of ordinary skill and used in

the specification. To be nested, Bellare's second cipher block needs to contain the first cipher block and a second plain text, the third cipher block to contain the second cipher block and a third plain text, and so forth. (See Applicant's Fig. 5).

Thus, Bellare does not remedy Aucsmith's deficiency in teachings. Therefore, claim 10 is patentable over Aucsmith, even when combined with Bellare.

Claims 11-13 and 17 depend on claim 10, incorporating its limitations. Therefore, for at least the same reasons, claims 11-13 and 17 are patentable over Aucsmith, even when combined with Bellare.

Claim 22, when entered, contains in substance the same "nested" limitation of claim 10. Therefore, for at least the same reasons, claim 22 is patentable over Aucsmith and Bellare combined.

Claim 23 depends on claim 22, incorporating its limitations. Therefore, for at least the same reasons, claims 11-13 and 17 are patentable over Aucsmith, even when combined with Bellare.

Claims 8-9 and 25-26 were also rejected in view of Aucsmith combined with Pendakur. Claims 8-9 and 25-26 depend on either claim 1 or 19, incorporating its limitations. Therefore, for at least the same reasons, claims 8-9 and 25-26 are patentable over Aucsmith.

Pendakur does not cure the earlier discussed deficiencies of Aucsmith. Therefore, claims 8-9 and 25-26 are patentable over Aucsmith even when combined with Pendakur.

Conclusion

In conclusion, remaining claim 1-2, 4-13, 17 and 19-27 are in condition of allowance. Early issuance of Notice of Allowance is respectfully requested.

Please charge any fees required for this submission or credit any overages to Deposit Account 500393.

Respectfully submitted,
Schwabe, Williamson and Wyatt

Date: 9/8/04

Aloysius AuYeung, Reg. No. 35,432
Attorney for Applicant

Pacwest Center, Suites 1600-1900
1211 SW Fifth Avenue
Portland, Oregon 97204-3795
Ph: 503.222.9981